US-PAT-NO: 6275907

DOCUMENT-IDENTIFIER: US 6275907 B1
See'image for Certificate of Correction

TITLE: Reservation management in a non-uniform memory access (NUMA) data processing system

DATE-ISSUED: August 14, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Baumgartner; Yoanna	Austin	TX	N/A	N/A
Carpenter; Gary Dale	Pflugerville	TX	N/A	N/A
Dean; Mark Edward	Austin	TX	N/A	N/A
Elman; Anna	Austin	TX	N/A	N/A
Fields, Jr.; James Stephen	Austin	TX	N/A	N/A
Glasco; David Brian	Austin	TX	N/A	N/A

US-CL-CURRENT: 711/143, 711/119 , 711/120 , 711/121 , 711/122 , 711/144 , 711/145

ABSTRACT:

A non-uniform memory access (NUMA) computer system includes a plurality of processing nodes coupled to a node interconnect. The plurality of processing nodes include at least a remote processing node, which contains a processor having an associated cache hierarchy, and a home processing node. The home processing node includes a shared system memory containing a plurality of memory granules and a **coherence** directory that indicates possible **coherence** states of copies of memory granules among the plurality of memory granules that are stored within at least one processing node other than the home processing node. If the processor within the remote processing node has a reservation for a memory granule among the plurality of memory granules that is not resident

within the associated cache hierarchy, the **coherence** directory associates the memory granule with a **coherence** state indicating that the reserved memory granule may possibly be held non-exclusively at the remote processing node. In this manner, the **coherence** mechanism can be utilized to manage processor reservations even in cases in which a reserving processor's cache hierarchy does not hold a copy of the reserved memory granule.

19 Claims, 6 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 6

---------- KWIC ---------

Abstract Text - ABTX (1):

A non-uniform memory access (NUMA) computer system includes a plurality of processing nodes coupled to a node interconnect. The plurality of processing nodes include at least a remote processing node, which contains a processor having an associated cache hierarchy, and a home processing node. The home processing node includes a shared system memory containing a plurality of memory granules and a **coherence** directory that indicates possible **coherence** states of copies of memory granules among the plurality of memory granules that are stored within at least one processing node other than the home processing node. If the processor within the remote processing node has a reservation for a memory granule among the plurality of memory granules that is not resident within the associated cache hierarchy, the **coherence** directory associates the memory granule with a **coherence** state indicating that the reserved memory granule may possibly be held non-exclusively at the remote processing node. In this manner, the **coherence** mechanism can be utilized to manage processor reservations even in cases in which a reserving processor's cache hierarchy does not hold a copy of the reserved memory granule.

Brief Summary Text - BSTX (11):

In accordance with the present invention, a non-uniform memory access (NUMA) computer system includes a plurality of processing nodes coupled to a node interconnect. The plurality of processing nodes include at least a remote processing node, which contains a processor having an associated cache hierarchy, and a home processing node. The home processing node includes a shared system memory containing a plurality of memory granules (e.g., cache lines) and a **coherence** directory that indicates possible **coherence** states of copies of memory granules among the plurality of memory granules that are stored within at least one processing node other than the home processing node.

Brief Summary Text - BSTX (12):

In the course of executing instructions, the processor within the remote processing node may execute a load-reserve instruction, which causes a specified cache line to be loaded into the processor's cache hierarchy and a reservation for the cache line to be set within the processor. If the processor subsequently loads a number of cache lines that map to the same congruence class of the cache hierarchy that contains the reserved cache line, the reserved cache line may be castout through conventional cache line replacement. The reservation is not affected by the replacement of the reserved cache line. According to the present invention, if the processor within the remote processing node has a reservation for a cache line that is not resident within its associated cache hierarchy, the **coherence** directory at the home processing node associates the cache line with a **coherence** state indicating that the reserved cache line may possibly be held non-exclusively at the remote processing node. In this manner, the cache **coherence** mechanism can be utilized to manage processor reservations even in cases in which a reserving processor's cache hierarchy does not hold a copy of the reserved cache line.

Brief Summary Text - BSTX (13):

The **coherence** state of the reserved cache line is set to the non-exclusive (e.g., shared) state in response to a writeback transaction transmitted from the remote processing node to the home processing node. In an embodiment of the present invention in which all cache line reservations are made visible (i.e., a processor cannot obtain a "silent" reservation for a cache line resident in its cache hierarchy), the replacement of a reserved cache line causes the reserving processor to issue a writeback-kill transaction indicating that the **coherence** indicator at the home processing node should be updated to an invalid state. In response to receipt of such a writeback-kill transaction while the reservation is valid, a node controller at the remote processing node converts the writeback transaction to a writeback-clean transaction, thereby indicating that the **coherence** indicator should be updated to the non-exclusive (e.g., shared) state.

Detailed Description Text - DETX (4):

To support reservations, each processor 10 includes a reservation register 15, illustrated in the embodiment of FIG. 1 as a portion of cache hierarchy 14. Reservation register 15 preferably contains an address field for storing the physical address of a memory granule for which the associated processor 10 may have a reservation and a valid bit for indicating whether or not the processor 10 holds a reservation for the memory granule located at the physical address contained within the address field. Like conventional processors, a processor 10 obtains a reservation for a memory granule by executing a load-reserve instruction, which, in addition to loading a specified memory granule (or subset thereof) into a register in processor core 12 and its associated L1 cache, loads the physical address of the memory granule into the address field of reservation register 15 and sets the valid bit. A processor's reservation for a memory granule is cleared by a number of events, including execution by the reserving processor 10 of a load-reserve specifying a different memory granule, execution by the reserving processor 10 or another processor 10 of a cache operation that invalidates the copy of the memory granule held in cache hierarchy 14, and **snooping** a "read with intent to modify" (RWITM) request transaction on local interconnect 16 in which another processor 10 requests exclusive access to the reserved memory granule for the purposes of modifying it.

Detailed Description Text - DETX (5):

As shown, each of processing nodes 8a-8n further includes a respective node controller 20 coupled between local interconnect 16 and node interconnect 22. Each node controller 20 serves as a local agent for remote processing nodes 8 by performing at least two functions. First, each node controller 20 **snoops** the associated local interconnect 16 and facilitates the transmission of local communication transactions to remote processing nodes 8. Second, each node controller 20 **snoops** communication transactions on node interconnect 22 and masters relevant communication transactions on the associated local interconnect 16. Communication on each local interconnect 16 is controlled by an arbiter 24. Arbiters 24 regulate access to local interconnects 16 based on bus request signals generated by processors 10 and compile **coherency** responses for **snooped** communication transactions on local interconnects 16, as discussed further below.

Detailed Description Text - DETX (9):

For purposes of the present discussion, the processing node 8 that stores a particular datum in its system memory 18 is said to be the **home node** for that datum; conversely, others of processing nodes 8a-8n are said to be remote nodes with respect to the particular datum.

Detailed Description Text - DETX (10):

Memory **Coherency**

Detailed Description Text - DETX (11):

Because data stored within each system memory 18 can be requested, accessed, and modified by any processor 10 within NUMA computer system 6, NUMA computer system 6 implements a cache **coherence** protocol to maintain **coherence** both between caches in the same processing node and between caches in different processing nodes. Thus, NUMA computer system 6 is properly classified as a CC-NUMA computer system. The cache **coherence** protocol that is implemented is implementation-dependent and may comprise, for example, the well-known Modified, Exclusive, Shared, Invalid (**MESI**) protocol or a variant thereof. Hereafter, it will be assumed that cache hierarchies 14 and arbiters 24 implement the conventional **MESI** protocol, of which node controllers 20 recognize the M, S and I states and consider the E state to be merged into the M state for correctness. That is, node controllers 20 assume that data held exclusively by a remote cache has been modified, whether or not the data has actually been modified.

Detailed Description Text - DETX (13):

Local interconnects 16 and node interconnect 22 can each be implemented with any bus-based broadcast architecture, switch-based broadcast architecture, or switch-based non-broadcast architecture. However, in a preferred embodiment, at least node interconnect 22 is implemented as a switch-based non-broadcast interconnect governed by the 6xx communication protocol developed by IBM Corporation. Local interconnects 16 and node interconnect 22 permit split transactions, meaning that no fixed timing relationship exists between the address and data tenures comprising a communication transaction and that data packets can be ordered differently than the associated address packets. The utilization of local interconnects 16 and node interconnect 22 is also preferably enhanced by pipelining communication transactions, which permits a subsequent communication transaction to be sourced prior to the master of a previous communication transaction receiving **coherency** responses from each recipient.

Detailed Description Text - DETX (14):

Regardless of the type or types of interconnect architecture that are implemented, at least three types of "packets" (packet being used here generically to refer to a discrete unit of information)--address, data, and

coherency response--are utilized to convey information between processing nodes 8 via node interconnect 22 and between **snoopers** via local interconnects 16. Referring now to Tables I and II, a summary of relevant fields and definitions are given for address and data packets, respectively.

Detailed Description Text - DETX (15):
As indicated in Tables I and II, to permit a recipient node or **snooper** to determine the communication transaction to which each packet belongs, each packet in a communication transaction is identified with a transaction tag. Those skilled in the art will appreciate that additional flow control logic and associated flow control signals may be utilized to regulate the utilization of the finite communication resources.

Detailed Description Text - DETX (16):
Within each processing node 8, **coherency** responses are communicated between each **snooper** and the local arbiter 24. The signal lines within local interconnects 16 that are utilized for **coherency** communication are summarized below in Table III.

Detailed Description Text - DETX (17):
Coherency responses transmitted via the AResp and AStat lines of local interconnects 16 preferably have a fixed but programmable timing relationship with the associated address packets. For example, the AStatOut votes, which provide a preliminary indication of the response of each **snooper** to an address packet on local interconnect 16, may be required in the second cycle following receipt of the address packet. Arbiter 24 compiles the AStatOut votes and then issues the AStatIn vote a fixed but programmable number of cycles later (e.g., 1 cycle). Possible AStat votes are summarized below in Table IV.

Detailed Description Text - DETX (18):
Following the AStatIn period, the ARespOut votes may then be required a fixed but programmable number of cycles (e.g., 2 cycles) later. Arbiter 24 also compiles the ARespOut votes of each **snooper** and delivers an

ARespIn vote, preferably during the next cycle. The possible AResp votes preferably include the **coherency** responses listed in Table V.

Detailed Description Text - DETX (19):

The ReRun AResp vote, which is usually issued by a node controller 20, indicates that the **snooped** request has a long latency and that the source of the request will be instructed to reissue the transaction at a later time. Thus, in contrast to a Retry AResp vote, a ReRun makes the recipient of a transaction that voted ReRun (and not the originator of the transaction) responsible for causing the communication transaction to be reissued at a later time.

Detailed Description Text - DETX (21):

Referring now to FIG. 2, there is illustrated a more detailed block diagram of a node controller 20 in NUMA computer system 6 of FIG. 1. As shown in FIG. 2, each node controller 20, which is coupled between a local interconnect 16 and node interconnect 22, includes a transaction receive unit (TRU) 40, a transaction send unit (TSU) 42, a data receive unit (DRU) 44, and a data send unit (DSU) 46. TRU 40, TSU 42, DRU 44 and DSU 46 can be implemented, for example, with field programmable gate arrays (FPGAs) or application specific integrated circuits (ASICs). As indicated, the address and data paths through node controller 20 are bifurcated, with address (and **coherency**) packets being processed by TRU 40 and TSU 42 and data packets being processed by DSU 44 and DRU 46.

Detailed Description Text - DETX (22):

TRU 40, which is so designated to indicate transaction flow off of node interconnect 22, is responsible for accepting address and **coherency** packets from node interconnect 22, issuing transactions on local interconnect 16, and forwarding responses to TSU 42. TRU 40 includes response multiplexer (mux) 52, which receives packets from node interconnect 22 and passes selected packets to both bus master 54 and **coherency** response logic 56 within TSU 42. In response to receipt of a address packet from response multiplexer 52, bus master 54 can initiate a communication transaction on its local interconnect 16 that is the same as or different from the type of communication transaction indicated by the received address packet.

Detailed Description Text - DETX (23):

TSU 42, which as indicated by its nomenclature is a conduit for transactions flowing onto node interconnect 22, includes a multiple-entry pending buffer 60 that temporarily stores attributes of communication transactions sourced onto node interconnect 22 that have yet to be completed. The transaction attributes stored in an entry of pending buffer 60 preferably include at least the address (including tag) of the transaction, the type of the transaction, and the number of expected **coherency** responses. Each pending buffer entry has an associated status, which can be set either to Null, indicating that the pending buffer entry can be deleted after all expected responses have been received, or to ReRun, indicating that the transaction is still pending. In addition to sourcing address packets on node interconnect 22, TSU 42 interacts with TRU 40 to process memory request transactions and issues commands to DRU 44 and DSU 46 to control the transfer of data between local interconnect 16 and node interconnect 22. TSU 42 also implements the selected (i.e., MSI) **coherency** protocol for node interconnect 22 with **coherency** response logic 56 and maintains **coherence** directory 50 with directory control logic 58.

Detailed Description Text - DETX (24):

Coherence directory 50 stores indications of the system memory addresses of data (e.g., cache lines) checked out to caches in remote nodes for which the local processing node is the **home node**. The address indication for each cache line is stored in association with an identifier of each remote processing node having a copy of the cache line and the **coherency** status of the cache line at each such remote processing node. Possible **coherency** states for

entries in
coherency directory 50 are summarized in Table VI.

Detailed Description Text - DETX (25):

As indicated in Table VI, the knowledge of the **coherency** s states of cache lines held by remote processing nodes is imprecise. This imprecision is due to the fact that a cache line held remotely can make a transition from S to I, from E to I, or from E to M without notifying the node controller 20 of the **home node**. In accordance with the present invention and as explained in greater detail below, the **coherency** states maintained in **coherence** directory 50 are also utilized to manage reservations in NUMA computer system 6.

Detailed Description Text - DETX (29):

As illustrated, the process begins at block 80 and thereafter proceeds to block 82, which illustrates a processor 10 initiating execution of a load-reserve instruction. As indicated at block 84, if the processor 10 that executes the load-reserve instruction (hereinafter, called reserving processor 10) is located within the **home node** of the cache line targeted by the load-reserve instruction, the process passes to blocks 90-104; alternatively, if the **home node** of the cache line targeted by the load-reserve instruction is a remote processing node 8, the process passes to blocks 120-150.

Detailed Description Text - DETX (30):

Referring now to block 90, execution of the load-reserve instruction entails a number of steps. First, processor core 12 of reserving processor 10 requests the cache line specified by the load-reserve instruction from its cache hierarchy 14, which may in turn place a request for the cache line on local interconnect 16 if the request for the cache line misses in cache hierarchy 14.
If the requested cache line does not reside in the cache hierarchy 14 associated with reserving processor 10 and a request transaction is accordingly issued on local interconnect 16, all **snoopers** coupled to local interconnect 16 issue a **coherency** response to the request transaction, as detailed above. In order to determine an appropriate **coherency** response to the request transaction by node controller 20, TSU 42 determines if **coherence** directory 50

indicates that the requested cache line is checked out to a remote processing node 8 in modified state. If not, the requested cache line can be obtained locally. However, if **coherence** directory 50 indicates the requested cache line is checked out to a remote processing node 8 in Modified state, TSU 42 replies to the request transaction on local interconnect 16 with a AResp ReRun **coherency** response and issues a request for the cache line to the remote processing node 8 via node interconnect 22. The remote processing node 8 will respond to the request by supplying the requested cache line to node controller 20, which will source the requested cache line to reserving processor 10 in response to the request transaction being reissued on local interconnect 16. Regardless of whether the requested cache line is obtained locally or from a remote processing node 8, once the requested cache line is supplied to cache hierarchy 14 of reserving processor 10, the requested cache line (or a subset thereof) is then loaded into a register within processor core 12. In addition, the base address of the cache line is stored within the address field of the requesting processor's reservation register 15, and the valid bit is set to indicate that the processor has a reservation for the requested cache line.

Detailed Description Text - DETX (31):

Following execution of the load-reserve instruction at block 90, the process proceeds to block 92, which illustrates a determination of whether or not reserving processor 10 has detected an event that would cancel the reservation. As noted above, the events that will cancel a reservation can include the execution by reserving processor 10 of a load-reserve targeting a different (i.e., unreserved) cache line, **snooping** a store or other transaction (e.g., read with intent to modify (RWITM)) on local interconnect 16 that modifies or indicates an intent to modify the reserved cache line, and execution by reserving processor 10 or a remote processor 10 of certain cache line invalidating instructions targeting the reserved cache line. If a reservation cancelling event is detected by reserving processor 10, the process proceeds to block 94, which depicts reserving processor 10 cancelling its reservation by resetting the valid bit within it s reservation register 15. The process then passes to block 100. if, on the other hand, a determination is made at block 92 that a reservation cancelling event has not been detected, the process proceeds directly to block 100.

Detailed Description Text - DETX (33):
Referring now to block 120, in the event that the cache line targeted by a load-reserve instruction has a remote processing node 8 as its **home node,** the processor 10 executing the load-reserve instruction requests the cache line specified by the load-reserve instruction from its cache hierarchy 14. If the cache hierarchy 14 cannot service the request, cache hierarchy 14 transmits a request for the cache line on local interconnect 16. If another of the local cache hierarchies 14 holds a copy of the requested cache line as indicated by the **coherency** responses received by the request transaction, then the data is supplied by the other cache hierarchy 14 by shared or modified intervention. If, however, the request cannot be serviced locally, the local node controller 20 provides an AResp ReRun **coherency** response to the request transaction and forwards the request transaction to the requested cache line's **home node** via node interconnect 22. The node controller 20 at the **home node** responds to the request transaction by obtaining the requested cache line from its local system memory 18, a local cache hierarchy 14, or a remote processing node 8. Once the requested cache line is obtained by the node controller 20 of the **home node,** DSU 46 within the **home node's** node controller 20 transmits a data packet containing the requested cache line to the node controller 20 of the **requesting node,** which in turn transmits the requested cache line to reserving processor 10 in response to the request transaction being reissued on local interconnect 16. Regardless of which of the foregoing scenarios is utilized to supply the requested cache line to the requesting processor 10, the **home node's**

coherence directory 50 will indicate the **coherence** state of the requested cache line as Shared or Modified at the processing node 8 containing reserving processor 10. In addition, the requested cache line (or a subset thereof) will be loaded into a register within processor core 12 of reserving processor 10, the address of the cache line will be loaded in the address field of the reserving processor's reservation register 15, and the valid bit of reservation register 15 will be set to indicate that reserving processor 10 has a valid reservation.

Detailed Description Text - DETX (35):

Block 130 depicts a determination of whether or not the cache line reserved at block 120 is to be castout of cache hierarchy 14 of the reserving processor 10, for example, due to the operation of the cache line replacement policy (e.g., LRU) of cache hierarchy 14. If not, the process proceeds to block 142, which is described below. However, if the cache line reserved at block 120 is being castout of cache hierarchy 14 of reserving processor 10, the **coherence** state of the cache line within the reserving processor's cache hierarchy 14 is updated to Invalid, as depicted at block 131. A determination is also made at block 132 whether or not the cache line was held by the cache in Modified or Exclusive state. If not, no further action is required to castout the cache line, and the process passes through page connector A to block 142. If, however, the castout cache line was held in Modified state or Exclusive state by cache hierarchy 14 of reserving processor 10, the cache line will be written back to the **home node's** system memory 18, as shown at blocks 134 and 136.

Detailed Description Text - DETX (36):

Block 134 depicts cache hierarchy 14 of reserving processor 10 writing back the castout cache line by transmitting a Writeback-clean transaction with the cache line to the cache line's **home node** via local interconnect 16, the local node controller 20, and node interconnect 22. In response to receipt of the

Writeback-clean transaction, the **home node's** node controller 20 updates the **home node's** system memory 18 with the castout cache line (which may or may not be modified), as depicted at block 136. In addition, the **home node's** node controller 20 updates the **coherence** state of the modified cache line in the **home node's coherence** directory 50 from Modified to Shared. By transitioning from Modified to Shared state rather than Modified to Invalid (as would be done for a Writeback-kill), **coherence** directory 50 retains an imprecise, conservative indication that a processor 10 at a remote processing node 8 may still retain a reservation for the modified cache line that was written back to the **home node**. Thus, when the **home node's** node controller 20 **snoops** a transaction that should cancel the reserving processor's reservation (which are the same set of transactions that would invalidate a remote cache line), node controller 20 of the **home node** will forward the transaction to the processing node 8 containing the reserving processor 8 in accordance with the **coherence** protocol, thereby cancelling the reservation, if any, as discussed above with respect to blocks 122 and 124. As a result of this conservative approach to reservation management, reservation correctness if guaranteed, albeit at the expense of unnecessary bus traffic in scenarios in which transactions are forwarded to remote processing nodes that no longer maintain a reservation.

Detailed Description Text - DETX (39):

Referring now to FIG. 4, there is depicted a high level logical flowchart of a second illustrative embodiment of a method for managing reservations in a NUMA computer system in accordance with the present invention. As illustrated, the process begins at block 180 and thereafter proceeds to block 182, which depicts a reserving processor 10 initiating execution of a load-reserve instruction. As indicated at block 184, if reserving processor 10 is located within the **home node** of the cache line targeted by the load-reserve instruction, the process passes to blocks 190-204; alternatively, if the **home node** of the cache line targeted by the load-reserve instruction is a remote processing node 8, the process passes to blocks 220-250.

Detailed Description Text - DETX (40):

Referring now to block 190, execution of the load-reserve instruction entails a number of steps. First, processor core 12 of reserving processor 10 requests the cache line specified by the load-reserve instruction from its cache hierarchy 14. Cache hierarchy 14 responds to the request by supplying the cache line to processor core 12, as discussed above with respect to block 90 of FIG. 3. Regardless of whether the requested cache line is obtained locally or from a remote processing node 8, once the requested cache line is supplied to reserving processor 10, the requested cache line (or a subset thereof) is then loaded into a register within processor core 12. In addition, the base address of the cache line is stored within the address field of the requesting processor's reservation register 15, and the valid bit is set to indicate that the processor has a reservation for the requested cache line. If obtaining the cache line for which a reservation is sought did not entail sourcing a request transaction on local interconnect 16 (i.e., the requested cache line was resident in cache hierarchy 14), reserving processor 10 also issues a load-reserve transaction on local interconnect 16 to ensure visibility of the reservation. In response to **snooping** the load-reserve transaction, TSU 42 of local node controller 20 enters the reservation in local reservation table 62. The reservation preferably indicates both the processor having the reservation and the base physical address of the reserved cache line.

Detailed Description Text - DETX (41):

Following execution of the load-reserve instruction at block 190, the process proceeds to block 192, which illustrates a determination of whether or not reserving processor 10 has detected an event that would cancel the reservation. If a reservation cancelling event is detected by reserving processor 10, the process proceeds to block 194, which depicts reserving processor 10 cancelling its reservation by resetting the valid bit within its reservation register 15. In addition, the reservation recorded within local reservation table 62 is cancelled either in response to node controller 20 **snooping** a reservation cancelling transaction on local interconnect 16 or in response to a transaction issued on local interconnect 16 by reserving processor 10. The process then passes to block 200. If, on the other hand, a determination is made at block 192 that a reservation cancelling event has not been detected, the process proceeds directly to block 200.

Detailed Description Text - DETX (43):

Referring again to block 184, in the event that the cache line targeted by a load-reserve instruction has a remote processing node 8 as its **home node** the process passes to block 220. Block 220 depicts the execution of the load-reserve by reserving processor 10, which as described above entails reserving processor 10 requesting the cache line specified by the load-reserve instruction from its cache hierarchy 14. Cache hierarchy 14 responds to the request by supplying the requested cache line to reserving processor 10, possibly after initiating transactions on local interconnect 16 and/or node interconnect 22, as discussed above with respect to block 120 of FIG. 3. Regardless of how the requested cache line is supplied to the requesting processor 10, the **home node's coherence** directory 50 will indicate the **coherence** state of the requested cache line as Shared or Modified at the processing node 8 containing reserving processor 10. In addition, the requested cache line (or a subset thereof) will be loaded into a register within processor core 12 of reserving processor 10, the address of the cache line will be loaded in the address field of the reserving processor's reservation register 15, and the valid bit of reservation register 15 will be set to indicate that reserving processor 10 has a valid reservation. In addition, the reservation will be recorded in local reservation table 62, as discussed above with respect to block 190.

Detailed Description Text - DETX (45):

Block 230 depicts a determination of whether or not the cache line reserved at block 220 is to be castout of cache hierarchy 14 of the reserving processor 10, for example, due to the operation of the cache line replacement policy (e.g., LRU) of cache hierarchy 14. If not, the process proceeds to block 242, which is described below. However, if the cache line reserved at block 220 is being castout of cache hierarchy 14 of reserving processor 10, the **coherence** state of the cache line within the reserving processor's cache hierarchy 14 is updated to Invalid, as depicted at block 231. A determination is also made at block 232 whether or not the castout cache line was modified. If not, no further action is required to castout the cache line, and the process passes to block 242. If, however, the castout cache line was held in Modified state by cache hierarchy 14 of reserving processor 10, the process proceeds to block 234, which depicts reserving processor 10 transmitting a Writeback-kill transaction with the modified cache line to the local node controller 20 via local interconnect 16. As illustrated at block 233, a determination is then made by TSU 42 of local node controller 20 whether or not local reservation table 62 indicates that reserving processor 10 has a valid reservation for the castout cache line. If the reserving processor's reservation for the modified cache line has been cancelled at block 224, local node controller 20 transmits the Writeback-kill transaction and the modified cache line to the **home node's** node controller 20 via node interconnect 22, as shown at block 237. In response to receipt of the Writeback-kill transaction, node controller 20 at the **home node** updates the **home node's** system memory 18 with the modified cache line data, as depicted at block 238. In addition, the **home node's** node controller 20 updates the **coherency** state of the cache line within the **home node's coherence** directory 50 from Modified to Invalid. The process then passes through page connector B to block 242.

Detailed Description Text - DETX (46):

Returning to block 233, if TSU 42 of local node controller 20 determines that the reserving processor's reservation for the castout cache line is still valid, the process passes to block 235, which depicts TSU 42 of local node controller 20 converting the Writeback-kill transaction received from reserving processor 10 into a Writeback-clean transaction and transmitting the Writeback-clean transaction with the modified cache line to the **home node's** node controller 20 via node interconnect 22. As illustrated at block 236, in response to receipt of the Writeback-clean transaction, the **home node's** node controller 20 updates the **home node's** system memory 18 with the modified cache line. In addition, the **home node's** node controller 20 updates the **coherence** state of the modified cache line in the **home node's coherence** directory 50 from Modified to Shared. As discussed above, the Shared state in **coherence** directory 50 will cause the **home node's** node controller 20 to forward **snooped** transactions to the processing node 8 containing reserving processor 10 in accordance with the cache **coherence** protocol. In this manner, forwarded transactions that would invalidate the reserved cache line at reserving processor 10 serve to cancel the reservation maintained by reserving processor 10. The process then passes from block 236 to block 242.

Detailed Description Text - DETX (48):

As has been described, the present invention provides an improved method and system for reservation management in a NUMA computer system. In accordance with the present invention, each processing node's **coherence** directory maintains indications of the possible **coherency** states of cache lines checked out from the processing node's system memory to cache hierarchies in other processing nodes. In order to keep track of possibly valid reservations for modified cache lines castout from remote cache hierarchies, the **coherence** directory updates the **coherence** state of each castout cache line written back from a remote processing node with a Writeback-clean transaction from Modified to Shared state. In this manner, transactions that require the cancellation of reservations, which are the same set of transaction that would require the invalidation of remote copies of a cache line according to the cache **coherence** protocol, are forwarded from the **home node** to all remote processing nodes containing processors holding reservations for the targeted cache lines. In response to **snooping** such forwarded transactions, the processors holding reservations for the relevant cache lines, if any, each cancel their respective reservation. In each of the disclosed embodiments, reservations in a NUMA computer system are appropriately cancelled in all cases. In a first embodiment, maintaining reservation correctness at all times comes at the minor expense of unnecessary traffic on the node interconnect in a statistically small number of processing scenarios in which the **home node** forwards a transaction to a remote processing node to cancel a reservation that has already been cancelled. In the second embodiment, such unnecessary traffic is eliminated by incorporating additional logic in the node controller that manages the writeback of castout cache lines to the **home node**.

Detailed Description Paragraph Table - DETL (1):

TABLE I Field Name Description Address Modifiers defining attributes of a <0:7> communication transaction for **coherency,** write thru, and protection Address Tag used to identify all packets within a <8:15> communication transaction Address Address portion that indicates the <16:63> physical, virtual or I/O address in a request AParity Indicates parity for address bits <0:63> <0:2> TDescriptors Indicate size and type of communication transaction

Detailed Description Paragraph Table - DETL (2):

TABLE I Field Name Description Address Modifiers defining attributes of a <0:7> communication transaction for **coherency,** write thru, and protection Address Tag used to identify all packets within a <8:15> communication transaction Address Address portion that indicates the <16:63> physical, virtual or I/O address in a request AParity Indicates parity for address bits <0:63> <0:2> TDescriptors Indicate size and type of communication transaction

Detailed Description Paragraph Table - DETL (3):

TABLE III Signal Name Description AStatOut Encoded signals asserted by each bus <0:1> receiver to indicate flow control or error information

to arbiter AStatIn Encoded signals asserted by arbiter in <0:1> response to tallying the AStatOut signals asserted by the bus receivers
ARespOut Encoded signals asserted by each bus <0:2> receiver to indicate
coherency information to arbiter ARespIn Encoded signals asserted by arbiter
in <0:2> response to tallying the ARespOut signals asserted by the bus
receivers

Detailed Description Paragraph Table - DETL (4):
TABLE IV AStat vote Meaning Null Idle Ack Transaction accepted by
snooper Error Parity error detected in transaction Retry Retry transaction,
usually for flow control

Detailed Description Paragraph Table - DETL (5):
TABLE V **Coherency** responses Meaning Retry Source of request must retry
transaction -- usually for flow control reasons Modified Line is modified in
cache and will be intervention sourced to requestor Shared Line is held
shared in cache Null Line is invalid in cache ReRun **Snooped** request has long
latency and source of request will be instructed to reissue transaction at a
later time

Detailed Description Paragraph Table - DETL (6):
TABLE VI Possible Possible **Coherence** state(s) state(s) directory in
local in remote state cache cache Meaning Modified I M,E, or Cache line may
be (M) I modified at a remote node with respect to system memory at **home**
node Shared S or I S or I Cache line may be held (S) non-exclusively at
remote node Invalid M,E,S, I Cache line is not held (I) or I by any remote
node Pending- S or I S or I Cache line is in the shared process of being
invalidated at remote nodes Pending- I M,E, or Cache line, which may
modified I be modified remotely, is in process of being written back to
system memory at **home node,** possibly with invalidation at remote node

Claims Text - CLTX (1):
1. A method of reservation management in a multiprocessor computer

system including a remote processing node and a home processing node coupled to said node interconnect, wherein said remote processing node includes a processor having an associated cache hierarchy and said home processing node includes a shared system memory containing a plurality of memory granules and a **coherence** directory that indicates possible **coherence** states of remote copies of said plurality of mimory granules, said method comprising:

Claims Text - CLTX (3):
while said mimory granule is not resident within said associated cache hierarchy, setting a **coherence** indicator within said **coherence** directory of said home processing node to a state indicating that said reserved memory granule is possibly held non-exclusively at said remote processing node, such that cache **coherence** communication will reset said reservation if necessary.

Claims Text - CLTX (7):
setting said **coherence** indicator to a state indicating that said memory granule is held only at said remote processing node; and

Claims Text - CLTX (9):
3. The method of claim 2, wherein writing back said memory granule comprises transmitting a writeback transaction from said remote processing node to said home processing node, wherein said writeback transaction indicates that said **coherence** indicator should be updated to shared state.

Claims Text - CLTX (11):
transmitting a writeback transaction to said node controller that indicates that said **coherence** indicator should be updated to invalid state; and

Claims Text - CLTX (12):
in response to receipt of said writeback transaction at said node controller, converting said writeback transaction prior to transmission of said writeback transaction to said home processing node, wherein said converted writeback transaction indicates that said **coherence** indicator should be updated

to shared state.

Claims Text - CLTX (17):
updating said **coherence** indicator to an invalid state.

Claims Text - CLTX (18):
7. The method of claim 1, wherein setting said **coherence** indicator comprises setting said state on response to a writeback of said reserved memory granule to said home processing node by said remote processing node.

Claims Text - CLTX (23):
a **coherence** directory that indicates possible **coherence** states of copies of memory granules among said plurality of memory granules that are stored within at least one processing node other than said home processing node;

Claims Text - CLTX (24):
a controller that, while said processor has a reservation for a memory granule among said plurality of memory granules that is not resident within said cache hierarchy, sets a **coherence** indicator to a state indicating that said reserved memory granule is possibly held non-exclusively at said remote processing node, such that cache **coherence** communication will reset said reservation if necessary.

Claims Text - CLTX (25):
9. The computer system of claim 8, wherein said controller updates said **coherence** indicator from a state indicating that said memory granule is held only at said remote processing node to said state indicating that said reserved memory granule is possibly held non-exclusively at said remote processing node in response to receipt from said remote processing node of a writeback transaction specifying said memory granule.

Claims Text - CLTX (26):
10. The computer system of claim 9, wherein said writeback transaction indicates that said **coherence** indicator should be updated to shared state.

Claims Text - CLTX (27):

11. The computer system of claim 9, said remote processing node further including a node controller that receives said writeback transaction from said processor, said writeback transaction indicating that said **coherence** indicator should be updated to invalid state, wherein said node controller converts said writeback transaction prior to transmitting said writeback transaction to said home processing node, such that said converted writeback transaction indicates that said **coherence** indicator should be updated to shared state.

Claims Text - CLTX (32):

wherein said controller updates said **coherence** indicator to an invalid state in response to receipt of a transaction at said home processing node indicating an update to said memory granule.

Claims Text - CLTX (35):

a **coherence** directory that indicates possible **coherence** states of copies of memory granules among said plurality of memory granules that are stored within at least one processing node other than said home processing node; and

Claims Text - CLTX (36):

a controller that, while said processor has a reservation for a memory granule among said plurality of memory granules that is not resident within said cache hierarchy, sets a **coherence** indicator to a state indicating that said reserved memory granule is possibly held non-exclusively at said remote processing node, such that cache **coherence** communication will reset said reservation if necessary.

Claims Text - CLTX (37):

15. The home processing node of claim 14, wherein said controller updates said **coherence** indicator from a state indicating that said memory granule is held only at said remote processing node to said state indicating that said reserved memory granule is possibly held non-exclusively at said remote processing node in response to receipt from said remote processing node of a

writeback transaction specifying said memory granule.

Claims Text - CLTX (38):
16. The home processing node of claim 15, wherein said writeback transaction indicates that said **coherence** indicator should be updated to shared state.

Claims Text - CLTX (41):
means for updating said **coherence** indicator from said state indicating that said reserved memory granule is possibly held non-exclusively by said remote processing node to an invalid state in response to receipt of a transaction at said home processing node indicating an update to said memory granule.

Claims Text - CLTX (42):
18. A remote processing node for a multi-node computer system, wherein said multi-node computer system contains a home processing node including a shared system memory containing a plurality of memory granules and a **coherence** directory that indicates possible **coherence** states of copies of memory granules that are stored within at least one processing node other than said home processing node, said remote processing node comprising:

Claims Text - CLTX (44):
a node controller that receives a writeback transaction from said processor, said writeback transaction indicating that a **coherence** indicator in said home processing node associated with a memory granule should be updated to invalid state, wherein said node controller converts said writeback transaction prior to transmitting said writeback transaction to said home processing node, such that said converted writeback transaction indicates that said **coherence** indicator should be updated to shared state.